Tool Development and Support

SPARK Toolset Release Note–Release 7.31

EXM/RN
Issue: 1.1
Status: Definitive
13 April 2006

Originator

SPARK Team

Approver

SPARK Team Line Manager

Copyright

The contents of this manual are the subject of copyright and all rights in it are reserved. The manual may not be copied, in whole or in part, without the written consent of Praxis High Integrity Systems Limited.

The software tools referred to in this manual are the subject of copyright and all rights in them are reserved. The rights in these tools are owned by Praxis High Integrity Systems Limited, and it may not be copied, in whole or in part, without the written consent of this company, except for reasonable back-up purposes. The same proprietary and copyright notices must be affixed to any permitted copies as were affixed to the original. This exception does not allow copies to be made for others, whether or not sold, and none of the material purchased may be sold, given or loaned to another person or organisation. Under law copying includes translating into another language or format.

©1991-2006 Praxis High Integrity Systems Limited, 20 Manvers St, Bath BA1 1PX

Limited Warranty

Praxis High Integrity Systems Limited save as required by law makes no warranty or representation, either express or implied, with respect to this software, its quality, performance, merchantability or fitness for a purpose. As a result, the licence to use this software is sold ‘as is’ and you, the purchaser, are assuming the entire risk as to its quality and performance.

Praxis High Integrity Systems Limited accepts no liability for direct, indirect, special or consequential damages nor any other legal liability whatsoever and howsoever arising resulting from any defect in the software or its documentation, even if advised of the possibility of such damages. In particular Praxis High Integrity Systems Limited accepts no liability for any programs or data stored or processed using Praxis High Integrity Systems Limited products, including the costs of recovering such programs or data.

SPADE is a trademark of Praxis High Integrity Systems Limited

Note: The SPARK programming language is not sponsored by or affiliated with SPARC International Inc. and is not based on SPARC™ architecture.

Contents

1 Introduction 5

2 Contact Information 6

3 SPARK language changes 7

4 Examiner changes 8
4.1 Correction to flow-analysis of array element parameters (SEPR 1962) 8
4.2 Examiner table sizes 9

5 SPADE Simplifier 10
5.1 Port to Mac PowerPC/OS X 10
5.2 New "/typecheck" option 10
5.3 Stricter checking of user-defined proof rules 10

6 SPADE Proof Checker 11
6.1 Correction to side-conditions in ARITH.RUL 11

7 POGS 12

8 SPARKFormat 13

9 SPARKMake 14

10 Backward incompatibilities 15

11 User manual change summary 16
11.1 Checker Rules 16
11.2 Examiner User Manual 16
11.3 Simplifier User Manual 16

12 Limitations and known errors 17
12.1 Tool limitations 17
12.2 Known error summary 19

13 Change Summary from Release 2.0 20
13.1 Release 2.0 - November 1995 20
13.2 Release 2.1 - July 1996 20
13.3 Release 2.5 - March 1997 20
13.4 Release 3.0 - September 1997 20
13.5 Release 4.0 - December 1998 21
13.6 Release 5.0 - June 2000 21
13.7 Release 6.0 - November 2001 22
13.8 Release 6.1 - June 2002 22
13.9 Release 6.3 – December 2002 24
13.10 Release 7.0 – July 2003 24
13.11 Release 7.2 – December 2004 24

13.12 Release 7.3 – February 2006 25

14 Operating system compatibility 27
14.1 VAX/VMS 27
14.2 SPARC/Solaris 27
14.3 Windows NT, 2000 and XP 27
14.4 Intel/Linux 27
14.5 Apple PowerPC/OS X 27

Document Control and References 28
Changes history 28
Changes forecast 28
Document references 28
File under 28

1 Introduction

Release 7.3 of the SPARK toolset was made available to users in February 2006. Subsequently, a defect was discovered in the Examiner's flow-analyser that necessitates a release 7.31 to correct this issue. A number of other small improvements have also been made to the Simplifier and Checker.

This document describes changes in the behaviour of all variants of the SPARK Toolset Release 7.31 compared to Release 7.3.

2 Contact Information

For further information about this document please contact Praxis High Integrity Systems:

By phone: +44 (0)1225 823829 (direct line), +44 (0)1225 466991 (exchange)

By FAX: +44 (0)1225 469006

By email: sparkinfo@praxis-his.com

3 SPARK language changes

None.

4 Examiner changes

This section documents other significant changes to the SPARK Examiner made for release 7.31. Where appropriate, SPARK Examiner Performance Report (SEPR) numbers are given.

4.1 Correction to flow-analysis of array element parameters (SEPR 1962)

Release 7.1 of the SPARK language and Examiner introduced a rule that allows array elements to be passed to formal procedure parameters of mode “in out” or “out”.

Subsequent analysis has shown that information flow analysis is not correct in the following case:

- The index of the array element passed is selected by an expression which includes variables or imported parameters, AND
- The formal parameter is mode “out”

In this case, the computed flow relation incorrectly excludes the variables used in the index expression.

SPARK Examiner releases affected by this defect are: 7.1, 7.2 and 7.3

Examiners prior to release 7.1 are not affected, since passing array elements as parameters was not allowed at all.

Example. Given:

```
type Index is range 1 .. 10;
type A is array (Index) of T;

procedure Clear (X : out T);
--# derives X from ;

S : A;
```

and:

```
procedure Clear_Elem (E : in Index)
--# global in out S;
--# derives S from S, E; -- correct
is
begin
   Clear (S (E));
end Clear_Elem;
```

Examiner 7.3 incorrectly reports:

```
!!! Flow Error : 30: The variable E is imported but neither
    referenced nor exported.

!!! Flow Error : 50: S is not derived from the imported
    value(s) of E.
```

This analysis is corrected in Examiner release 7.31.

Recommendation:

Projects using Examiners 7.1, 7.2 or 7.3 should upgrade to release 7.31 as soon as possible. A "back-to-back" analysis of programs should be performed to detect any instances of this problem.

Code that is normally analysed with Examiners 7.0 or earlier cannot be affected by this problem.

4.2 Examiner table sizes

The maximum sizes of the compilation unit and source file tables have been increased in the "Large" and "Mega" variants of the Examiner for release 7.31.

5 SPADE Simplifier

Simplifier 2.24 ships with toolset release 7.31. This incorporates the following improvements and features:

5.1 Port to Mac PowerPC/OS X

The "FreeDemo" version of the Simplifier that accompanies the "SPARK Book" is now available for the Apple Mac PowerPC/OS X operating system.

5.2 New "/typecheck" option

A new command-line option "/typecheck" is now available. This instructs the Simplifier to read in and type-check the input files (e.g. VCG, FDL and RLS files) but to perform no further simplification. No output files are produced.

This option is useful to quickly scan a large set of VCG files for legality before committing to a full simplification attempt, when used in conjunction with the SPARKSimp tool. For example, to scan a full directory tree of VCG files for type-correctness on Windows:

```
sparksimp /a /l /sargs /typecheck
```

Or on UNIX platforms:

```
sparksimp –a –l –sargs /typecheck
```

More details in the Simplifier User Manual, section 5.1.

5.3 Stricter checking of user-defined proof rules

Simplifier 2.22 could sometimes allow a user-defined proof rule to be applied incorrectly. In some circumstances, this could lead to simplifier mistakenly "proving" a VC that was actually not True. This has been corrected in release 2.24.

If a project is using Simplifier 2.22 and user-defined proof rules ("rlu files"), then we recommend that proofs should be re-generated and checked.

6 SPADE Proof Checker

SPADE Proof Checker version 2.07 ships with toolset release 7.31. This includes a single improvement over and above version 2.06:

6.1 Correction to side-conditions in ARITH.RUL

Two rules (intdiv(5) and intdiv(6)) have been corrected so that their side-conditions can be successfully evaluated. In prior versions of the Checker, these rules were unusable.

7 POGS

No change between releases 7.3 and 7.31.

8 SPARKFormat

No change between releases 7.3 and 7.31.

9 SPARKMake

No change between releases 7.3 and 7.31.

10 Backward incompatibilities

There are no known backward incompatibilities introduced between Examiner Releases 7.3 and 7.31.

11 User manual change summary

No changes, other than the addition of this release note, and:

11.1 Checker Rules

- Correction to rules in ARITH.RUL as described in section 6.1 above.

11.2 Examiner User Manual

- Version number updated to 7.31.

11.3 Simplifier User Manual

- Documentation of /typecheck option.

12 Limitations and known errors

12.1 Tool limitations

This section describes limitations of the Examiner tool arising mainly from incomplete implementation of planned features. Where appropriate a SPARK Examiner Performance Report (SEPR) number is given.

12.1.1 General

1 The SPARK 95 language definition removes the distinction between initial and later declarative items; this distinction remains in force in the Examiner that requires SPARK 83 declaration orders even in SPARK 95 mode. (SEPR 813)

2 The Examiner does not yet permit the use of 8-bit characters in SPARK 95 user-defined identifiers. (SEPR 818)

3 Universal expressions in a modular context may sometimes require type qualification. (SEPR 1591)

4 The Examiner does not yet permit the use of “use type” following an embedded package specification. (SEPR 747)

5 The Examiner does not yet permit the renaming of packages in the same way that subprograms can be renamed. (SEPR 1391)

6 The Examiner does not yet allow the ‘Base attribute when not used as a prefix. (SEPR 1114)

7 The Examiner does not yet allow S’Range where S is scalar. (SEPR 1115)

12.1.2 Verification Condition Generation and Run-time Checks

1 Ada string inequality is not modelled. (SEPR 712)

2 VCs involving string catenation that includes the character ASCII.NUL will be incorrect. (SEPR 661)

3 Aggregates of multi-dimensional arrays cannot be modelled although aggregates of arrays of arrays can. (SEPR 590)

4 Verification conditions involving real numbers are evaluated using infinite precision or perfect arithmetic; this allows the correctness of an algorithm to be shown but cannot guard against the effects of cumulative rounding errors for example.

5 The Examiner does not generate VCs for package initialization parts. Statically determinable constraint errors will be detected during well-formation checking of package initialization. (SEPR 288)

6 The VC Generator cannot model the implementation-dependent attributes of floating and fixed-point types; see section 12.1.3.

12.1.3 Attribute limitations

12.1.3.1 Unimplemented attributes

The following attributes are officially supported by SPARK according to the language definition, but are not yet implemented by the Examiner. The Examiner will generate error number 30 ("Attribute XXX is not yet implemented in the Examiner") if you try to use them.

- Adjacent
- Compose
- Copy_Sign
- Leading_Part
- Remainder
- Scaling
- Exponent
- Fraction
- Machine
- Model
- Rounding
- Truncation
- Unbiased_Rounding

Note that these are all function-like attributes concerning floating- and fixed-point types.

12.1.3.2 Unevaluable attributes

The Machine_* and Model_* attributes are accepted by the Examiner, but it does not know how to statically evaluate them since they are inherently implementation dependent. For example, the package:

```
package F is
   type T is digits 6 range –10.0 .. 10.0;
```

```
   C : constant := T'Machine_Emax;
end F;
```

is legal SPARK, but the Examiner does not know the actual numeric value of C.

12.2 Known error summary

This section lists known errors in the Examiner that are awaiting investigation and correction.

1 The SPARK rule that array actual parameters must have the same bounds as the formal parameter is not checked for function parameters where the actual parameter is a subtype of an unconstrained array type. Since subtype bounds are static in SPARK errors of this kind should be detected by an Ada compiler. If not an unconditional run-time error will occur. (SEPR 1060)

2 The Examiner permits the body of a subprogram to be entirely made up of proof statements thus breaching the Ada rule that at least one Ada statement must be present. (SEPR 278)

3 Where a package declares two or more private types the Examiner permits mutual recursion between their definitions in the private part of the package. (SEPR 848)

4 The Examiner does not take due account of a range constraint when determining the subtype of a loop variable; this affects completeness checking of case statements within the loop. For example `for I in Integer range 1..4 loop` would require only values 1, 2, 3 and 4 to be covered by the case statement. (SEPR 693)

5 When summarising the counts of pragmas found during an analysis the totals may depend on whether units are selected via the command line (or metafile) or using the index mechanism. The difference affects only pragmas placed *between* program units and arises because placing a file name on the command line causes the entire *file* to be analysed whereas selecting it using indexes causes only the required *unit* to be read from the file. (SEPR 483)

13 Change Summary from Release 2.0

A release note detailing changes from the previous version accompanies each Examiner Release; this section simply summarises the various changes that have been made.

13.1 Release 2.0 - November 1995

Release 2.0 added:

- static expression evaluation;
- variable initialization at declaration;
- full-range scalar subtypes; and
- operator renaming in package specifications.

13.2 Release 2.1 - July 1996

Release 2.1 added:

- facilities for proof of absence of run-time errors

13.3 Release 2.5 - March 1997

Release 2.5 was distributed with “High Integrity Ada - the SPARK Approach” and provided initial facilities for SPARK 95

13.4 Release 3.0 - September 1997

Windows NT was supported for the first time with this release. Release 3.0 also added:

- additional SPARK 95 support;
- flow analysis of record fields;
- command line meta files;
- named numbers;
- unqualified string literals;
- moded global annotations; and

- optional information flow analysis.

13.5 Release 4.0 - December 1998

With Release 4.0 we upgraded all users to a single product standard supporting SPARK 83, SPARK 95 and analysis options up to an including proof facilities. New features were:

- full implementation of public and private child packages;
- default switch file; and
- provision of the INFORMED design document.

13.6 Release 5.0 - June 2000

- Enhanced proof support:
  I. facilities for proof of programs containing “abstract state”;
  II. addition of quantified expressions;
  III. proof rule generation for enumeration types;
  IV. identification of the kind and source of each VC;
  V. suppression of trivially true VCs;
  VI. Proof Obligation Summariser tool (POGS)
- Optional HTML output files with hyperlinks that can be “browsed” interactively
- Better support for common Ada file naming conventions
- User-selectable annotation character
- Improved suppression of analysis were results might otherwise be misleading
- Static expression evaluation in proof contexts
- Singleton enumeration types
- Revised SPARK_IO package
- Error numbering

13.7 Release 6.0 - November 2001

- Introduction of "external variables" to simplify modelling of the interactions between a SPARK program and its external environment.
- Addition of the "null derives" annotation to describe information flows which affect only the external environment.
- Introduction of modular types
- Use of loop labels in exit statements
- Use of global modes on function subprograms
- Extended support for predefined types such as Long_Integer
- Simplified run-time check generation for own variables
- Relaxation of need for mandatory type announcement of own variables
- Plain output option to simplify comparisons of Examiner output files
- Platform-independent switch files and metafiles
- Support for intentionally infinite loops
- Detection of own variables that can never be initialized
- Detection of unusable private types
- Extra refinement checks on global variables when performing data flow analysis
- Detection of unnecessary others clause in case statements
- Extensions to the POGS tool
- Improved error messages to distinguish different cases of variables which are "not declared or visible"
- Improved SPADE Simplifier Release 2.0
- New "SPARKSimp" Tool

13.8 Release 6.1 - June 2002

- Introduction of tagged types

- Introduction of type assertion annotations
- Introduction of modular subtypes
- Introduction of the configuration file
- Introduction of the `help` command line switch
- Demo Examiner now runs on Linux.
- VCG generation for inherited operations of tagged types
- Improved handling of null derives
- Attributes ‘Floor and ‘Ceiling implemented
- Detection of duplicate record fields
- Improved overflow checks on universal integer expressions
- Corrected handling of loop invariants in while loops
- Strengthened behaviour of /noecho option
- Trapping non-positive accuracy in real type declaration
- Recursion in meta-files and index-files
- Improved handling of Address clauses
- Improved handling of Import and Interface pragmas
- VCG Modelling of Boolean membership operators
- Simplification of common Integer inequalities
- Simplification of common enumerated inequalities
- Simplification of VCs involving quantified expressions
- Simplifier performance
- Checker has new built-in rule families: MODU LAR, BITWISE and ENUMERATION
- Proved by review option in POGS

13.9 Release 6.3 – December 2002

Release 6.3 of the toolset was constructed to accompany the textbook "High Integrity Software: The SPARK Approach to Safety and Security" by John Barnes, but was not delivered to other users. Its main new features were:

- Slight revision to the rules regarding the placement of tagged type declarations.
- Correction to the modelling of Boolean type membership operators in the verification conditions.
- Support for generating VCs that allow the verification of the Liskov Subsitution Principal (LSP) for tagged types and their operations.
- Dramatically improved performance of the Simplifier, particularly in the simplification of quantified expressions.

13.10 Release 7.0 – July 2003

Release 7.0 of the toolset comprised:

Examiner version 7.0

Simplifier version 2.12

Release 7.0 added:

- Ravenscar profile extensions to the language.
- Support for Ada.Interrupts and Ada.Real_Time in the configuration file.
- The new /noduration command line switch.
- VC generation for unconstrained formal parameters.
- Suppression of VC generation for illegal function bodies.
- New "SPARKFormat" tool.

13.11 Release 7.2 – December 2004

Release 7.2 of the toolset comprised:

Examiner version 7.2

Simplifier version 2.17

Release 7.2 added:

- Unconstrained string constants to the language.
- Instantiation of Unchecked_Conversion to the language.
- (Full) record subtypes to the language.
- Declaration of subprograms in the private part of packages.
- Refined proof annotations for private types.
- % suffix for referring to value of variable on entry to loop in proof contexts.
- Extra hypotheses for local variables.
- Suppression of VC generation for illegal function bodies.
- Replacement rules for composite constants.
- Concurrent simplification with SPARKSimp.
- Improved simplification of VCs with large structured objects.
- Improved simplification of arithmetic and logical expressions.
- New “SPARKMake” tool.

13.12 Release 7.3 – February 2006

Release 7.3 of the toolset comprised:

Examiner version 7.3

Simplifier 2.22

Checker 2.06

Significant features of this release included:

- Improved VC Generation.
- New “error explanations” switch for Examiner.
- Generation of proof rules for ‘Size.
- Improved diagnosis and reporting of common syntax errors.

- Error and warning count summary in Examiner output.
- Use of pragma Import to complete an external own variable.
- Correction to FDL declaration order for private and announced types.
- New Simplifier tactics for dealing with rational inequalities and Examiner-generated proof rules.
- User-defined proof rules for the Simplifier.
- New Checker rules for MODULAR expressions.
- Significant performance improvement for Simplifier and Checker.

14 Operating system compatibility

14.1 VAX/VMS

Examiner 7.31 is available for VAX/VMS releases 5.5-2 and above. The other tools are not available for VAX/VMS at this time.

14.2 SPARC/Solaris

The toolset is compatible with Solaris 5.6 through to 10 including those with a 64-bit kernel.

14.3 Windows NT, 2000 and XP

The toolset is compatible with Windows NT 4.0, Windows 2000, and Windows XP. The executables are also known to work on Windows 95 and 98; however, use of the toolset on these operating systems is unsupported. The FlexLM licence manager software only runs on Windows NT, Windows 2000, or Windows XP.

14.4 Intel/Linux

All the toolset, with the exception of the Checker, is compatible with Intel-based Linux operating systems. Only the "FreeDemo" version of the toolset is currently available for Linux to support buyers of John Barnes' "SPARK Book." If you require a full professional SPARK toolset for Linux, then please contact us.

14.5 Apple PowerPC/OS X

All the toolset, with the exception of the Checker, is compatible with Apple's PowerPC OS X/Darwin operating systems. Only the "FreeDemo" version of the toolset is currently available for OS X to support buyers of John Barnes' "SPARK Book." If you require a full professional SPARK toolset for OS X, then please contact us.

Document Control and References

Praxis High Integrity Systems Limited, 20 Manvers Street, Bath BA1 1PX, UK.
Copyright © Praxis High Integrity Systems Limited 2006. All rights reserved.

Changes history

Issue 0.1 (30th March 2006): First draft for release 7.31.

Issue 1.0 (30th March 2006): Definitive issue following review.

Issue 1.1 (13th April 2006): Addition of Simplifier 2.24.

Changes forecast

Updates following review.

Document references

File under

CVSROOT/userdocs/Examiner_RN_7p31.doc